Module 7 – Transactions

Overview

The main purpose of this lab is to familiarise yourself with transactions within Microsoft SQL Server.

- Review the original code and identify possible issues
- Throw errors instead of printing an error message
- Catch an error from within code

Objectives

At the end of this lab, you will be able to:

- Declare an error within the sys.messages system table
- Raise an previously declared error
- Throw an error
- Produce code that catches errors and deals with the error efficiently

Setup: Launch SQL Server Management Studio

1. On the Start menu, click All Programs, click Microsoft SQL Server 2014 and then click SQL Server Management Studio.
2. The Microsoft SQL Server Management Studio window opens, and then the Connect to Server dialog box will appear.
3. In the Connect to Server dialog box, click Connect to accept the default settings.
4. On the toolbar, click New Query, and either select the QATSQLPLUS database in the Available Databases box, or type USE QATSQLPLUS in the query window.
5. If you wish, you may save your queries to My Documents or the desktop. All modules are separate and you will not require any queries from this module in any later module.

Exercise 1: Identifying possible issues

In this exercise, you will review the code and table constraints to determine possible issues that may arise.

The main tasks for this exercise are as follows:

1. Review the code and data constraints.
2. Identify possible issues.

Task 1: Design and code

1. The two tables used in this exercise are:
   - dbo.BookTransfers
     - ProductID int not null
     - TransferDate datetime not null
     - TransferReason varchar(30) not null
     - TransferAmount int not null
   - dbo.BookStock
     - ProductID not null
     - StockAmount int not null must be >= 0
2. The original code:

```
INSERT INTO dbo.BookTransfers VALUES (@ProductID,getdate(),'Transfer Out',@Amount)
UPDATE dbo.BookStock
    SET StockAmount = StockAmount - @Amount
    WHERE ProductID = @ProductID
```

3. What issues may arise from the code given the table designs?

Task 2: Alter the code to stop issues

1. Open the start code from “s:\qatsqlplus\M07 E01 Transactions.sql”.
2. Update the code to:
   - Check and reject NULL variable values
   - Start a TRY block
   - Start a transaction within the TRY block
   - Perform the original insert and update
   - Commit the transaction
   - End the TRY block
   - Start a CATCH block
   - Rollback the transaction
   - THROW an error
   - End the CATCH block
3. Test the query with different parameters:
   - @ProductID = 1, @Amount = 3000
     - The result should be that an error will be thrown
   - @ProductID = 1, @Amount = 1
     - The result should be that an error will be thrown
   - @ProductID = NULL, @Amount = 3
     - The result should be that an error will be thrown
   - @ProductID = 4, @Amount = 1
     - 2 “(1 row(s) affected)” messages
4. The query can be saved if you want.

Answers

The answers below are for example only. Coding style and order of columns should not matter.

Ex 1 Task 1	Possible issues: • All columns do not allow NULLS, so any variables should be checked for NULL • The StockAmount value must be greater or equal to zero, so during the UPDATE the calculation StockAmount - @Amount may cause a negative • Insert may succeed and the Update fail, leading to unbalanced stock figures

Ex 1 Task 2

```sql
-- TASK 2:
DECLARE @ProductID INT = 1
DECLARE @Amount INT = 20

IF (@ProductID IS NULL OR @Amount IS NULL)
  BEGIN;
    THROW 59999, 'Neither variable is allowed to be NULL',1
    RETURN
  END

BEGIN TRY
  BEGIN TRAN
    INSERT INTO dbo.BookTransfers VALUES
        (@ProductID,getdate(),'Transfer Out',-@Amount)
    UPDATE dbo.BookStock
      SET StockAmount = StockAmount - @Amount
      WHERE ProductID = @ProductID
  COMMIT TRAN
END TRY
BEGIN CATCH
  ROLLBACK TRAN;
  THROW 59999,'An error occurred in the transaction.  Everything rolled
back',1
END CATCH
GO
```